DRAWINGS

Enclosed please find two sheets of drawings, one annotated sheet with the changes made and one replacement sheet with amendments.

REMARKS

Claims 1-15, 17-23 and 25-28 are pending in the application prior to entering this amendment.

The examiner objects to the abstract of the disclosure because it is too long.

The examiner requests that applicants describe p-code.

The examiner rejects claims 1-15, 17-23, and 25-28 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter that the applicants regard as the invention. The examiner rejects claims 1, 2, 5-9, 11, and 12 under 35 U.S.C. § 102(e) as being anticipated by Arora et al. (U.S. Patent No. 6,625,693). The examiner rejects claims 1-4 are rejected under 35 U.S.C. §102(b) as being anticipated by Crump et al. (U.S. Patent No. 5,557,759).

The applicants amend claims 1, 10, 12-14, 25, and 28.

Claims 1-15, 17-23, and 25-28 remain in the application after entering this amendment.

The applicants add no new matter and request reconsideration.

Specification

The term p-code is well known in the art to refer to pseudo code.

Claim Rejections Under § 112

The examiner rejects claim 1 because it is unclear what is meant by "concurrent in-line stating." The applicants amend claim 1 to clarify that the term should read —concurrent in-line staging— to thereby obviate the examiner's rejection.

The examiner rejects claims 1 and 28 because it is unclear what is meant by "concurrent in-line staging." The limitation refers to the insertion of interrupt servicing instructions done such that they intervene the mainline program instructions within the queue mechanism thereby allocating core processor bandwidth between inserted interrupt servicing instructions and mainline program instructions.

The examiner rejects claim 14 because it is unclear what was meant by p-code and retiring back instructions. P-code is a well known abbreviation for pseudo code. The applicants amend claim 14 to make clear that *executed* instructions are retired back into the instruction queue.

The examiner rejects claim 22 because the utility of an impending natural context switch is unclear. The applicants respectfully traverse this rejection because no such rejection exists under § 112. The applicants seek the examiner's counsel.

The examiner rejects claim 25 because the origin of the micro-opcodes representing service instructions is ambiguous. The applicants amend claim 25 to clarify such origin.

The examiner rejects claim 28 because "the executed micro-opcodes" lacks antecedent basis. The applicants amend claim 28 to obviate the examiner's rejection.

The examiner appears to have only rejected claims 10, 13-15, 17-23, and 25-28 under §112. The applicants assume claims 10, 13-15, 17-23, and 25-28 are in condition for the examiner's allowance provided the applicants rewrite claims 10 and 13 in independent form including the corresponding base and intervening claims. The applicants have so rewritten claims 10 and 13.

Claim Rejections Under § 102

Anticipation requires "the presence in a single prior art disclosure of all elements of a claimed invention arranged as in that claim." *Panduit Corp. v. Dennison Manufacturing Co.*, 774 F.2d 1082, 1101, 227 U.S.P.Q. 337, 350 (Fed. Cir. 1985) (quoting *Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1548, 220 U.S.P.Q. 193, 198 (Fed. Cir. 1983)).

I. Arora Does Not Anticipate the Current Invention

The examiner rejects claims 1, 2, 5-9, 11, and 12 as old over Arora. The applicants respectfully disagree for the reasons that follow.

Claim 1 recites *detecting an interrupt service request*. The examiner alleges Arora's disclosure of "instructions to be executed [that] are put into a pool and the processor executes them without regard to programmer-specified order" (column 3, lines 34-36) makes old the recited detecting. But this passage does not appear to disclose detecting, much less detecting an interrupt signal as required by the claim. And, in Arora, the "dynamic handler determiner *tracks the occurrence* of exceptions" and defines an exception as "any unpredictable event." (Column 3, lines 49-50, emphasis added and column 2, lines 64-65). An unpredictable event does not disclose an interrupt service request.

Claim 1 recites *inserting interrupt servicing instruction into an instruction queue mechanism*. The examiner appears to allege that Arora's lightweight handler discloses the recited interrupt servicing instruction and the splice cache 101 discloses the instruction queue mechanism. But Arora's lightweight handlers are "microcoded for handling an exception and

the splice cache 101 is "a block of memory containing the plurality of lightweight handlers" (column 2, lines 62-64). In Arora, the lightweight handlers are not interrupt servicing instructions in that they handle the exception rather than being the instruction itself. Claim 1 requires not that the fetch portion of the instruction queue mechanism be inserted into the cache portion, but rather the interrupt servicing instruction itself is inserted in the queue mechanism. As such, Arora does not disclose inserting interrupt servicing instructions into an instruction queue mechanism.

Claim 1 recites *processor bandwidth being allocated between the inserted interrupt servicing instructions and other program instructions.* The interrupt servicing instructions as inserted into the instruction queue mechanism result in bandwidth being allocated. The examiner alleges that because Arora's processor executes instructions sequentially that it then discloses allocating bandwidth. But, Arora's processor makes no purposeful allocation of bandwidth, and certainly no allocation of bandwidth between interrupt service instructions and other program instructions as recited. To do so would require Arora to disclose that its lightweight hander being inserted into the splice cache results in bandwidth being allocated between the lightweight handler and other program instructions. It does not.

Claim 1 recites *executing the interrupt servicing instructions and other program instructions.* The examiner alleges Arora's element 105 teaches the recited executing. Arora describes its element 105 as a pipeline where instructions are stored, not executed (column 3, lines 13-14, 34-36). Because Arora element 105 does not execute any instructions, much less interrupt servicing instructions and other program instructions as required by claim 1.

Claim 2 recites fetching instructions from an instruction fetch unit and decoding instructions into micro-opcodes, and Arora fails to teach an instruction fetch unit or a decoder. The examiner does not cite to any part of Arora to show that it anticipates these elements of the current invention. As such, Arora cannot anticipate an instruction fetch unit or a decoder.

Claim 2 recites *fetch instructions from the instruction cache, the processing being performed in such manner as to decode the instructions into micro-opcodes and execute the micro-opcodes in one or more out-of-order execution units.* The applicants acknowledge that Arora discloses a fast exception processing system that may include an out-of-order machine. But it is difficult to understand how Arora's out-of-order machine discloses the recited limitations of claim 2. ("Another way for injecting handlers may be used for out-of-order machines." column. 3, line 31). Nowhere does Arora disclose micro-opcodes, for example,

much less decoding the fetched instructions into micro-opcodes and executing the micro-opcodes in one or more out-of-order execution units.

Finally, claims 5-9 recite *detecting plural interrupts by prioritizing the plural interrupts and inserting one or more instances of the interrupt servicing instructions into the instruction queue mechanism in accordance with one or more predefined interrupt servicing allocation criteria*. Arora fails to teach any prioritization of exceptions. The applicants acknowledge that Arora discloses instructions that are put in a pool and executed without regard to the programmer-specified order, but nowhere does Arora teach prioritization of the interrupts as recited. In addition, Arora teaches that "the most frequently occurring exceptions [are put] into the slice cache" (column 3, lines 54-55). Arora discloses a frequently occurring exception, which is a single exception that occurs more than once over time, but not detecting plural interrupts --occurring at a single point in time-- as recited.

II. Crump Does Not Anticipate the Current Invention

The examiner rejects claims 1-4 as old over Crump. The applicants disagree for the reasons that follow.

Claim 1 recites *detecting an interrupt service request*. Crump fails to disclose *detection* of an interrupt. Crump "*incorporates* an interrupt vector request register, which *receives and retains* the appropriate vector address upon occurrence of an interrupt" (column 18, lines 20-22, emphasis added).

Claim 1 recites *inserting interrupt servicing instructions into an instruction queue ... where inserting the interrupt servicing instructions into the instruction queue mechanism results in processor bandwidth being allocated between the inserted interrupt servicing instructions and other program instructions via concurrent in-line staging of the interrupt servicing instructions and other program instruction*. In contrast, Crump *fetches* a code on a miss (column 18, line 24).

As stated above, claim 1 recites *bandwidth being allocated between the inserted interrupt servicing instructions and other program instructions*. The examiner asserts that Crump "executes instruction in sequence, thus allocating bandwidth" (Office Action, December 23, 2005). For the same reasons listed for distinguishing Arora above, Crump does not disclose the invention's method of bandwidth-allocation.

Because Crump fails to teach detection of an interrupt service request, an inserting an interrupt instruction, or bandwidth-allocation Crump does not anticipates these elements of the current invention.

Finally, claims 2-4 recite *the processing being performed in such manner as to decode the instructions into micro-opcodes and execute the micro-opcodes in one or more out-of-order execution units.* While Crump's processor is micro-coded, nowhere does Crump disclose processing as recited, i.e., processing in such a manner as to decode the instructions into micro-opcodes and execute micro-opcodes in one or more out-of-order execution units. In addition, Crump executes in sequence as the examiner has previously acknowledged (Office Action, December 23, 2005, page 5). The current invention "interrupt service demands are met with more or less resistance based not fixes upon criteria or underlying assumptions" (Application, page 19, lines 9-10). Because Crump fails to teach a manner to decode the instructions and discloses executing in order, Crump does not anticipate claims 2-4.

The applicants request the examiner to remove his §102 rejections and allow claims 1-9 and 11-12.
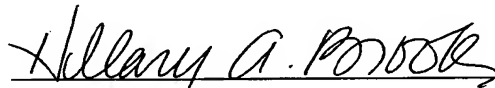
Conclusion

The applicants request reconsideration and allowance of all remaining claims. The applicants encourage the examiner to telephone the undersigned at (503) 222-3613 if it appears that an interview would be helpful in advancing the case.

Customer No. 32231

Respectfully submitted,

MARGER JOHNSON & McCOLLOM, P.C.

Hillary A. Brooks
Reg. No. 45,815

MARGER JOHNSON & McCOLLOM, P.C.
210 SW Morrison Street, Suite 400
Portland, OR 97204
503-222-3613

Annotated Sheet Showing Changes

FIG. 1

10

12a →
12b →
12n →

INTERRUPT CONCENTRATOR

14

P

16

CORE PROCESSOR

INSTRUCTION QUEUE

28

20

FROM ~~TO~~ MEMORY

INSTRUCTION FETCH UNIT

FETCH-AND-DECODE UNIT

34

38

PREDICTION UNIT

18

P-CODE DECODER

32

ALLOCATION LOGIC

FROM MEMORY

INSTRUCTION CACHE

22

OOO EXECUTION UNIT

24

P-CODE RETIREMENT UNIT

26

DISPATCH-AND-EXECUTE UNIT

36